1.0	2.0	4.0	LABEL	DESCRIPTION
0000	0000	0000	usrpok	$4c constant = JMP
0001	0001	0001	addprc	USR vector - Lo/Hi
005a	0003	0003	strsiz	# of locations per $ descriptor
			integr	1 byte integer from "qint"
			charac	Starting delimiter
				(Search character, usually $00)
				(May have another use in Fat Forty only)
005b	0004	0004	endchr	Ending delimiter
			addpr4	
				(Scan between quotes flag)
005c	0005	0005	count	General counter
				(Input buffer pointer; # of subscripts)
005d	0006	0006	dimflg	Flag to remember dimensioned variables
005e	0007	0007	valtyp	" for variable type: 00=numeric; ff=string
005f	0008	0008	intflg	" " # type: 80=integer; 00=floating point
			addpr8	
0060	0009	0009	dores	" whether can crunch reserved words
				(Flag: DATA scan; LIST quote; memory)
0061	000a	000a	clmwid	Size of print window
			subflg	Flag which allows subscripts in syntax
				(FNX flag)
0062	000b	000b	inpflg	Flags input or read
				(00=INPUT; 40=GET; 98=READ)
0063	000c	000c	domask	Mask used by relation operations
				(Comparison Evaluation flag)
			tansgn	Flag sign of tangent
-	-	000d	dsdesc	ds$ length
0064	000d	-	outsup	(Flag to suppress output: +=prompt, -=suppress)
-	-	000e		(ds$ pointer - Lo/Hi)
0003	000e	0010	channl	Active I/O channel #
			chanl	
0008	0011	0011	poker	Holds address for poke command -Lo/Hi
				(Integer value for SYS, GOTO)
			linnum	Line # storage - Lo/Hi
?	0012	0012	forsiz	# of bytes used on stack for-next
0065	0013	0013	temppt	index to next available descriptor
0066	0014	0014	lastpt	Pointer to last $ temp - Lo/Hi
				(Pointers for descriptor stack)
0068	0016	0016	tempst	Storage for numtmp temp descriptors
				(Descriptor stack - 4 temp $ pointers - thru 1e)
?	0017	0017	numlev	# of GOSUB levels allowed
0071	001f	001f	indexl	Indirect index #1 - Lo/Hi
			index	
0073	0021	0021	index2	" " #2 - Lo/Hi
0075	0023	0023	resho	Res - register
				(Product area for multiplication - thru 27)
0076	0024	0024	resmoh	
0077	0025	0025	addend	Temp used by "umult"
			resmo	
0078	0026	0026	reslo	
007a	0028	0028	txttab	Pointer to start of BASIC text - Lo/Hi
007c	002a	002a	vartab	" " " " variables - Lo/Hi
007e	002c	002c	arytab	" " " " array table - Lo/Hi
0080	002e	002e	strend	" " end of variables - Lo/Hi
0082	0030	0030	fretop	" " start of actual strings - Lo/Hi
0084	0032	0032	frespc	" " top of free string space -Lo/Hi
0086	0034	0034	memsiz	Highest RAM address available for BASIC - Lo/Hi
0088	0036	0036	curlin	Current line # being executed - Lo/Hi

1.0	2.0	4.0	LABEL	DESCRIPTION
008a	0038	0038	oldlin	Last line # executed (for CONT command) - Lo/Hi
008c	003a	003a	oldtxt	Old txtptr (for CONT) & temp storage - Lo/Hi
008e	003c	003c	datlin	Current data line # (in case of errors) - Lo/Hi
0090	003e	003e	datptr	Data statement pointer -Lo/Hi
0092	0040	0040	inpptr	Source of input address - Lo/Hi
0094	0042	0042	varnam	Current variable name - 2 byte
0096	0044	0044	varpnt	Pointer to variable in memory - Lo/Hi
			fdecpt	Pointer into powers of 10 for "fout"
0098	0046	0046	lstpnt	" to list $ - Lo/Hi
			forpnt	" " current FOR-NEXT var. reference - Lo/Hi
009a	0048	0048	vartxt	Pointer into list of variables - Lo/Hi
			opptr	" to current operator in table - Lo/Hi
				(Y-save; op-save; BASIC pointer save)
009c	004a	004a	opmask	Mask created by current operator
				(Comparison symbol accumulator)
009d	004b	004b	grbpnt	Pointer used in garbage collection
			defpnt	" " " function definition
			tempf3	A third FAC temporary - 4 bytes
				(Misc work area, pointers - thru 50)
009f	004d	004d	dscpnt	Pointer to a string description
00a2	0050	0050	four6	Variable constant used by garb collect
00a3	0051	0051	jmper	4c = JMP: subroutine for dispatch of functions
00a4	0052	0052	size	address for dispatch of functions - Lo/Hi
00a5	0053	0053	oldov	The old overflow
00a6	0054	0054	tempf1	A FAC temp - 4 bytes
				(Misc numeric work area - thru 5d)
00a7	0055	0055	arypnt	A pointer used in array building
			highds	Destination of highest element in blt.
00a9	0057	0057	hightr	Source of highest element to move
00ab	0059	0059	tempf2	A FAC temp - 4 bytes
00ac	005a	005a	deccnt	# of places before decimal point
			lowds	Location of last byte transferred into
00ad	005b	005b	tenexp	Base 10 exponent for "fin" & "fout"
00ae	005c	005c	grbtop	Pointer used in garbage collection
			dptflg	Flag if a decimal point has been input
			lowtr	Last thing to move in blt.
00af	005d	005d	expsgn	Sign of base 10 exponent
			epsgn	
00b0	005e	005e	dsctmp	This is where temp descs are built
			fac	Main floating-point accumulator
				(Mantissa - thru 62)
			facexp	The exponent byte
00b1	005f	005f	facho	Most significant byte of mantissa
00b2	0060	0060	facmoh	One more
00b3	0061	0061	facmo	Middle order of mantissa
			indice	Indice is set up here by "qint"
00b4	0062	0062	faclo	Least significant byte of mantissa
00b5	0063	0063	facsgn	Sign of FAC, = 0 or -1 when unpacked
00b6	0064	0064	sgnflg	Sign of FAC is preserved here by "fin"
			degree	A constant used by polynomials
				(Series evaluation constant pointer)
00b7	0065	0065	bits	Counter for # of bit shifts to normalize FAC
				(FAC overflow)
00b8	0066	0066	argexp	Argument register exponent
				(Accum. #2 - thru 6b)
00b9	0067	0067	argho	
00ba	0068	0068	argmoh	
00bb	0069	0069	argmo	

1.0	2.0	4.0	LABEL	DESCRIPTION
00bc	006a	006a	arglo	
00bd	006b	006b	argsgn	The sign - same as FAC
00be	006c	006c	strngl	Pointer to a $ or descriptor
			arisgn	A sign reflecting the result (Sign comparison, Acc. #1 vs. #2)
00bf	006d	006d	facov	Overflow byte of the FAC (Accum. #1 lo-order - rounding)
00c0	006e	006e	bufptr	Pointer to buffer used by "crunch"-Lo/Hi
			strng2	" " $ or desc.
			polypt	" " polynomial coefficients
			fbufpt	" " "fbuffer" used in "fout"
			curtol	Absolute linear index is formed here (Cassette buff len/Series pointer)
00c2	0070	0070	chrget	Subroutine - gets next character from BASIC text (thru 87)
00c8	0076	0076	chrgot	Subroutine - regets current character from B. text
00c9	0077	0077	txtptr	Pointer to current BASIC source text - Lo/Hi
00d9	0087	0087	chrrts	End of chrget
00da	0088	0088	rndx	Next Random number - thru 8c
0200	008d	008d	ctimr	Jiffy (1/60th. sec.) clock for TI and TI$ - 3 bytes
			time	
0219	0090	0090	cinv	IRQ interrupt vector - Lo/Hi
021b	0092	0092	cbinv	BRK " " "
none	0094	0094	nminv	NMI " " "
020c	0096	0096	satus	Variable ST - I/O Status byte
0203	0097	0097	lstx	Which key down; 255=none
0204	0098	0098	sfst	Shift key: 1 if depressed
0205	0099	0099	crfac	Correction factor for clock - 2 bytes
0209	009b	009b	stkey	Copy of keyboard PIA @ e812: STOP and RVS flags
?	009c	009c	svxt	Timing constant for tape
0209	009d	009d	verck	Load=0, Verify=1
020d	009e	009e	ndx	Number of characters in keybd buffer
020e	009f	009f	rvs	Screen reverse flag
?	00a0	00a0	c3p0	IEEE output; 255=character pending
?	00a1	00a1	indx	(End-of-line-for-input pointer)
?0045	00a3	00a3	lsxp	Cursor row
?00E2	00a4	00a4	lstp	Cursor column
			totk	
?	00a5	00a5	bsour	(IEEE output buffer)
?0223	00a6	00a6	sfdx	Key image
?0224	00a7	00a7	blnsw	(0=flash cursor)
0225	00a8	00a8	blnct	Blink counter for cursor flash
0226	00a9	00a9	gdbln	Character under cursor
0227	00aa	00aa	blnon	Cursor in blink phase
?	00ab	00ab	syno	EOT received from tape
?	00ac	00ac	crsw	(Input from screen/from keyboard)
?	00ad	00ad	xsav	X save
0262	00ae	00ae	ldtnd	How many open files
0263	00af	00af	dfltn	Default input device, normally 0
0264	00b0	00b0	dflto	Default output device, normally 3
0265	00b1	00b1	prty	Tape character parity
?	00b2	00b2	dpsw	(Byte received flag)
?	00b3	00b3	wsw	(Logical Address temporary save)
0020	00b4	00b4	savx	Place to save X
			tl	(Tape buffer character; Monitor command)
0268	00b5	00b5	rcnt	(Temporary counter - File name pointer)
0021	00b5	00b5	tmpc	
			t2	

1.0	2.0	4.0	LABEL	DESCRIPTION
0022	00b6	00b6	tmpc2 t3	
?	00b7	00b7	pcntr	Serial bit count
?	00b8	00b8	ptch	
?	00b9	00b9	firt	(Cycle counter)
0270	00ba	00ba	cntdn	Tape writer countdown
0271	00bb	00bb	bufpt	Tape buffer pointers, #1 and #2 - Lo/Hi
0273	00bd	00bd	shcnl	(Write leader count; read pass1/2)
0274	00be	00be	rer	(Read error flag - Write new byte)
0275	00bf	00bf	rez	(Read bit seq error - Write start bit)
0276	00c0	00c0	ptr1	Error log pointers - pass 1
0277	00c1	00c1	ptr2	" " " - pass 2
0278	00c2	00c2	rdflg	0=Scan/1-15=Count/$40=Load/$80=End
0279	00c3	00c3	shcnh	Write leader length; read checksum
00e0	00c4	00c4	pnt	Pointer to screen line address - Lo
00e1	00c5	00c5	point	" " " " " - Hi
00e2	00c6	00c6	pntr	Column of cursor on screen line
00e3	00c7	00c7	sal	Start address Lo - pointer: tape, scroll
00e4	00c8	00c8	sah	" " Hi " " "
00e5	00c9	00c9	eal	End address Lo End of current program
00e6	00ca	00ca	eah	" " Hi " " " "
?	00cb	00cb	cmp0	(Tape timing constants - 2 bytes)
?	00cc	00cc	temp	
00ea	00cd	00cd	qtsw	(0=direct cursor, else programmed)
?	00ce	00ce	snswl	(Tape read timer 1 enabled)
000b	00cf	00cf	diff	(EOT received from tape)
?	00d0	00d0	prp xid	(Read character error)
00ee	00d1	00d1	fnlen xdl	File name length
00ef	00d2	00d2	la xd2	Current file logical address
00f0	00d3	00d3	sa	Current file secondary addrs
00f1	00d4	00d4	fa	First address - current file device number
00f2	00d5	00d5	lnmx	Right-hand window or line margin
00f3	00d6	00d6	tbuf	Pointer: Start of tape buffer - Lo/Hi
00f5	00d8	00d8	tblx	Active cursor line
00f6	00d9	00d9	data	Last key/checksum/misc.
00f7	00da	00da	fnadr	File name address pointer - Lo/Hi
?00FB	00dc	00dc	insrt	Number of INSERTs outstanding
?	00dd	00dd	ochar	Write shift word/read character in
000a	00de	00de	wrap fsblk	Wrap FFFF flag - Tape blocks remaining to write/read
?00FE	00df	00df	mych	Serial word buffer
0229	00e0	00e0	E76E 80col	40 column screen line wrap table - thru f8 (Lo of start address of $19 lines)
-	-	00e0	ldtbl xrec	Top, bottom of window - 2 bytes - 80 column.
		00e1	xwrt	
-	-	00e2		Left window margin - " "
-	-	00e3		Limit of keybd buffer - " "
-	-	00e4		Key repeat flag - " "
-	-	00e5		Repeat countdown - " "
-	-	00e6		New key marker - " "
-	-	00e7		Chime time - " "
-	-	00e8		HOME count - " "
-	-	00e9		Fat Forty unlike other 40 columns thru $f9
				Input vector - 2 bytes - 80 column

1.0	2.0	4.0	LABEL	DESCRIPTION
-	-	00eb		Output vector - Lo/Hi - " "
-	-	00f1	xfn1	- " "
-	-	00f2	xfn2	- " "
0207	00f9	00f9	cas1	Cassette #1 status
0208	00fa	00fa	cas2	" #2 "
00f7	00fb	00fb	sta1	Start tape address - Lo
0011	00fb	00fb	tmp0	Indirect 1 - Lo/Hi
00f8	00fc	00fc	stah	Start tape address - Hi
0013	00fd	00fd	tmp2	Indirect 2 - Lo/Hi
?	00fd	00fd	fnadr2	
		00fe		(DOS pointer, misc. - 2 bytes)
?	00ff	00ff	lofbuf	+ index = start of "fout" $ for "strd" & ti$
0100	0100	0100	fbuffr	"fout" buffer holds ASCII $ for output
			bad	Processor stack - thru 01ff
				(STR$ work area, Monitor work - thru 010a)
				(Tape read error log - thru 013e)
	01fb	01fb	stkend	Stack end for BASIC
	01ff	01ff	zz1	
			zz4	
			zz5	
000a	0200	0200	buf	Input buffer - thru 0250
001a	0200	0200	pch	PC - Hi
0019	0201	0201	pcl	PC - Lo
			zz2	
001b	0202	0202	flgs	Flags - 6502 ST register
			zz3	
001c	0203	0203	acc	Acc
001d	0204	0204	xr	X reg.
001e	0205	0205	yr	Y "
001f	0206	0206	sp	SP
-	0207	0207	invh	User IRQ - Hi
-	0208	0208	invl	" " Lo
?	0209	0209	savnam	
			zzz1	
0242	0251	0251	lat	File logical address table - thru 025a
024c	025b	025b	fat	File first address table - thru 0264
0256	0265	0265	sat	File secondary address table - thru 026e
020f	026f	026f	keyd	Keyboard input buffer - thru 0278
027a	027a	027a	tape1	Tape#1 input buffer - thru 0339
033a	033a	033a	tape2	" #2 " " " 03f9
-	-		fnlen2	DOS character pointer
			pos	
-	-	033b	drive1	DOS drive 1 flag
-	-	033c	drive2	DOS drive 2 flag
-	-	033d	lrecl	(DOS length/write flag)
			didchk	
-	-	033e	parchk	DOS parameter check - syntax flags
-	-	033f	diskid	DOS disk ID - 2 bytes
-	-	0341	count	DOS command string count
-	-	0342	tbuf2	DOS file name buffer - 10 bytes
-	-	0353	tbuff	DOS command string buffer - thru 0380
-	-	03e9		New key marker - Fat Forty - same as $e6 on CBM 4.0
-	-	03ea		Key repeat countdown - " " " " $e5 " " "
-	-	03eb		Keyboard buffer limit - " " " " $e3 " " "
-	-	03ec		Chime time - " " " " $e7 " " "
-	-	03ed		Decisecond timer - " " only
-	-	03ee		Key repeat flag - " " - same as $e4 on CBM 4.0
-	-	03ef		[tab] work value - " " only

1.0	2.0	4.0	LABEL	DESCRIPTION
-	-	03f0		PET Fat Forty tab stop table - thru $03f9
-	03ee	03ee		CBM model " " " " $03f7
-	03fa	03fa	usrcmd	Monitor extension vector - Lo/Hi
-	none	03fc	bob	IEEE timeout defeat
0400	0400	0400	ramloc	Available RAM for BASIC text - thru $7fff
8000	8000	8000	offset	Screen RAM - thru $83ff 40-col. / $87ff 80 col.
9000	9000	9000		Available ROM area - thru $afff 4.0 / $bfff 1.0 & 2.0
				4K bank-select RAM area in Superpet only
-	-	a000		4K system ROM - Superpet only
c000	c000	b000	romloc	First ROM location
			stmdsp	Start of instruction dispatch table
c046	c046	b066	fundsp	" " function " "
c04c	c04c	b06c	usrloc	USR instruction's jump address within function table
c074	c074	b094	optab	Start of math operators dispatch table
c089	c089	b0a9	negtab	Unitary negate dispatch - .by 7d
c08c	c08c	b0ac	nottab	Not operator dispatch - .by 5a
c08f	c08f	b0af	ptdorl	Comparison dispatch - .by 64
c089	c089	b0a9	negtab	Unitary negate dispatch - .by 7d, dispatch
c092	c092	b0b2	reslst	Start of reserved word list
				(Shift flags end of keyword, 00 flags end of table)
c190	c192	b20d	errtab	Start of BASIC error message storage
c28d	c28b	b306	err	Message - error
c294	c292	b30d	intxt	" - in
c299	c297	b312	reddy	" - ready.
c2a4	c2a2	b31b	brktxt	" - break
c2ac	c2aa	b322	fndfor	Peeks at the stack for an active "for" loop
c2b1	c2af	b327	ffloop	(Top of "for" peek loop)
c2c6	c2c4	b33c	cmpfor	(Compare stack to for/next variable pointer)
c2d2	c2d0	b348	addfrs	(No match, check further)
c2d9	c2d7	b34f	ffrts	(End of "for" peek)
c2da	c2d8	b350	bltu	Opens up a space in BASIC for a new line
c2e1	c2df	b357	bltuc	(Entry from d502-2.0)
c2fe	c2fc	b374	bltl	+
c30a	c308	b380	bltlp	+
c30e	c30c	b384	morenl	+
c315	c313	b38b	decblt	+
c31d	c31b	b393	getstk	Test if stack too deep - abort if is
c32a	c328	b3a0	reason	Checks for available memory space
c334	c332	b3aa	trymor	+
c338	c336	b3ae	reasav	+
c343	c341	b3b9	reasto	+
c356	c354	b3cc	rearts	(End of mem check)
c357	c355	b3cd	omerr	Preset for "Out of memory" and...
c359	c357	b3cf	error	Error handler - message index in X and...
c366	c364	b3da	errcrd	(Do "crlf" and...)
c36c	c36a	b3e0	geterr	(Read error message from table and...)
c379	c377	b3ed	typerr	Restore data pointer & Print "error" and...
c380	c37e	b3f4	errfin	(Print string - address in A & Y and...)
c38b	c389	b3ff	ready	Warm start of BASIC - NMI vector in 2.0 & 4.0
c394	c392	b406	main	Main BASIC loop - analyzes input lines
c3ac	c3ab	b41f	mainl	Lines that start with a no. handled here
c3e7	c3e6	b45a	qdectl	+
c3ef	c3ee	b462	mloop	+
c3fd	c3fc	b470	nodel	(Adjust pointers for cold start)
c418	c417	b48b	nodelc	+
c428	c431	b4a5	stolop	(Top of minor loop)
c430	c439	b4ad	fini	Cleans up BASIC system - CLR
c433	c442	b4b6	lnkprg	Relinks BASIC instructions in text area

1.0	2.0	4.0	LABEL	DESCRIPTION
c43c	c44b	b4bf	chead	+
c447	c453	b4c7	czloop	(Count til find zero)
-	c46e	b4e1	lnkrts	(End of link)
c468	c46f	b4e2	inlin	Receive line into input buffer - max. len 79+RETURN
c46a	c471	b4e4	inlinc	(get a character from the keyboard)
c476	c47e	b4f8	fininl	+
c48d	c495	b4fb	crunch	Looks up keywords in an input line
c493	c49b	b501	kloop	+
c49e	c4a7	b50d	cmpspc	(a SPACE?)
c4b4	c4bd	b523	kloopl	(a "0"?)
c4bc	c4c5	b52b	mustcr	+
c4c6	c4cf	b53d	reser	+
c4c8	c4d1	b544	rescon	+
c4da	c4e0	b552	getbpt	+
c4dc	c4e2	b554	stuffh	+
c4ef	c4f5	b567	colis	+
c4f1	c4f7	b569	nodatt	+
c4f8	c4fe	b570	strl	+
c500	c507	b579	strng	+
c507	c50e	b580	nthis	+
c50b	c512	b584	nthisl	+
-	-	b58d	nthis2	+
c51a	c522	b599	crdone	+
c522	c52c	b5a3	fndlin	Search for address of line whose # is in "linnum"
c526	c530	b5a7	fndlnc	(Entry from b84b)
c53d	c547	b5be	fndlol	+
c546	c550	b5c7	affrts	+
c54f	c559	b5d0	flinrt	(line # not found)
c550	c55a	b5d1	flnrts	(" " found)
c551	c55b	b5d2	scrath	NEW instruction - clears all pointers
c553	c55d	b5d4	scrtch	(Entry from d445)
c567	c572	b5e9	runc	(adjust pointers for cold start of BASIC)
c770	c577	b5ee	clear	CLR instruction - (clears variable pointers)
c56a	c579	b5f0	clearc	(Entry from b80d)
c581	c590	b60b	fload	(Restore data pointer)
c584	c593	b60e	stkini	(Entry from b3ed)
c588	c597	b612	(pop)	(Undo all gosubs & goto's - use before panic exit)
c597	c5a6	b621	stkrts	+
c59a	c5a7	b622	stxtpt	"txtptr"="txttab"-1
c5a8	c5b5	b630	list	LIST instruction
c5b0	c5bd	b638	golst	(Convert char. $ to # in 11-12)
c5c7	c5d4	b64f	lstend	+
c5d5	c5e2	b65d	list4	+
c5f2	c5ff	b67a	tstdun	(done?)
c5f4	c601	b67c	typlin	(Print the integer in A,X)
c5fb	c608	b683	prit4	+
c5ff	c60c	b687	ploop	(Print character in A)
c60c	c619	b694	ploopl	+
c61e	c62d	b6a8	grody	(Jump to "ready")
c621	c630	b6ab	qplop	+
c633	c642	b6c5	resrch	+
c636	c645	b6c8	rescrl	+
-	-	b6ce	rescr2	+
c63e	c64d	b6d4	prit3	+
-	-	b6d5	prit3b	+
c649	c658	b6de	for	FOR instruction
c65a	c669	b6ef	notol	+
c692	c6a1	b727	ldfone	(Continue to build FOR vectors)

1.0	2.0	4.0	LABEL	DESCRIPTION
c6a6	c6b5	b73b	oneon	(Extract FAC sign)
c6b5	c6c4	b74a	newstt	Read & execute next statement
-	c6d4	b759	dircon	+
-	c6e4	b769	dircnl	+
c6e9	c6f7	b77c	gone	Dispatches next byte "chrget" returns
c6f2	c700	b785	gone3	Dispatches A if <>0 else loop to "newstt"
c6f5	c702	b787	gone2	(Entry from b8e7)
-	c71a	b795	gone4	+
-	c717	b7a2	glet	(Jump to perform LET)
-	c71a	b7a5	morsts	(A ":"?)
c6cc	c71e	b7a9	snerrl	(Jump to print "syntax error")
-	c72a	b7ac	go	Handle GO token - find a TO
c70d	c730	b7b7	restor	RESTORE statement
c717	c73a	b7c1	resfin	(Entry from bce2)
c71b	c73e	b7c5	iscrts	(End of RESTORE)
c71c	c73f	b7c6	stop	STOP instruction if carry set - else...
			bstop	O.S. Substitute: "stop" is also a label at $f343
c71e	c741	b7c8	end	END instruction
c71f	c742	b7c9	stopc	(<>00?)
c72b	c751	b7d8	stpend	(Entry from bbf2)
c733	c759	b7e0	diris	+
c735	c75b	b7e2	endcon	(Entry from b766)
c742	c768	b7eb	gordy	Jump to "ready"
c745	c76b	b7ee	cont	CONT instruction
c75e	c784	b807	contrt	(End of CONT)
c775	c785	b808	run	RUN instruction
c780	c790	b813	gosub	GOSUB instruction
c794	c7a4	b827	runc2	(Entry from b810)
c79d	c7ad	b830	goto	GOTO instruction
c7b4	c7c4	b847	luk4it	+
c7b8	c7c8	b84b	lukall	+
c7c9	c7d9	b85c	gorts	(End of GOTO)
c7ca	c7da	b85d	return	RETURN instruction
c7db	c7eb	b86e	userr	Print "undefined instruction error"
c7e0	c7f0	b873	snerr2	Jump to "syntax error"
c7e3	c7f3	b876	retul	+
c7f0	c800	b883	data	(DATA instruction)
c7f3	c803	b886	addon	(Add Y to scan pointer)
c7fd	c80d	b890	remrts	(End of DATA)
c7fe	c80e	b891	datan	Scan for next ":"
c801	c811	b894	remn	" " " end-of-line - "txtptr" offset in Y
c809	c819	b89c	exchqt	+
c811	c821	b8a4	remer	+
c820	c830	b8b3	if	IF instruction
c82f	c83f	b8c2	okgoto	(Allow GOTO after IF)
c833	c843	b8c6	rem	REM instruction
c838	c848	b8cb	docond	+
c840	c850	b8d3	doco	(Jump to do instruction)
c843	c853	b8d6	ongoto	ON instruction
c84b	c85b	b8de	snerr3	(Must include GOTO or GOSUB - else "syntax error")
c84f	c85f	b8e2	onglop	+
c857	c867	b8ea	onglpl	(Get next char. & convert $ to #)
c862	c872	b8f5	ongrts	(End of ON)
c863	c873	b8f6	linget	Get integer from BASIC & put in "linnum"
c869	c879	b8fc	morlin	(Entry from b92d)
c897	c8a7	b92a	nxtlgc	(Get next char. from input buffer)
c89d	c8ad	b930	let	LET instruction
c8ba	c8ca	b94d	qintgr	(Entry from bc8a)

1.0	2.0	4.0	LABEL	DESCRIPTION
c8ce	c8de	b961	copflt	("fac" to FOR pointer)
c8d1	c8e1	b964	copstr	+
c8d2	c8e2	b965	inpcom	(Entry from bc7f)
c8e8	c8f5	b978	timelp	+
c902	c90f	b992	noml6	+
c912	c91f	b9a2	timest	+
c91c	c928	b9ab	timnum	+
c923	c92f	b9b2	fcerr2	Jump to "illegal quantity error"
c926	c932	b9b5	gotnum	+
c92b	c937	b9ba	getspt	Copy strings if needed
-	-	b9be	dskx0	+
-	-	b9d2	dskx1	+
-	-	b9d4	dskx2	+
c93c	c948	b9e1	qvaria	+
c94a	c956	b9ef	dntcpy	(Don't copy)
c951	c95d	b9f6	copy	(Do ")
c967	c973	ba13	copyc	(Entry from b9f3)
-	-	ba2e	copy00	+
-	-	ba44	copy01	+
-	-	ba46	copy02	+
-	-	ba4e	stradj	Point to string for a copy
-	-	ba6c	adj	+
-	-	ba70	adjxx	+
-	-	ba74	adj02	+
-	-	ba83	adj00	+
-	-	ba85	adj01	+
c97f	c98b	ba88	printn	PRINT# instruction
c985	c991	ba8e	cmd	CMD instruction
c98f	c99b	ba98	saveit	+
c999	c9a5	baa2	strdon	+
c99c	c9a8	baa5	newchr	(Get current character from buffer)
c99f	c9ab	baa8	print	PRINT instruction
c9a1	c9ad	baaa	printc	(Entry from bb15)
c9c8	c9d5	bad2	fininl	(Entry from b4f8)
c9d8	c9e2	badf	crdo	Print RETURN character
-	c9ec	baed	crfin	(Entry from bb34)
c9f8	c9ee	baef	prtrts	+
c9f9	c9ef	baf0	comprt	(Process "," to correct column)
c9fc	c9f2	baf3	morcol	+
ca06	c9fc	bafd	taber	TAB & SPC handler
-	ca0c	bb0d	aspac	+
-	ca0d	bb0e	xspac	+
-	ca0e	bb0f	xspac2	+
ca21	ca11	bb12	notabr	+
-	ca17	bb18	xspacl	(Jump to print CURSOR-RIGHT)
ca27	ca1c	bb1d	strout	Output string addressed in Y and A - end with 00
ca2a	ca1f	bb20	strprt	Print string pointed to by "index"
ca31	ca26	bb27	strpr2	(Entry from bb32)
-	ca39	bb3a	outspc	Output a SPACE
ca44	ca40	bb41	crtskp	Output CURSOR-RIGHT
ca47	ca43	bb44	outqst	Output a "?"
ca49	ca45	bb46	outdo	Output character in A
ca74	ca4c	bb49	outrts	(Force ST register bits)
ca77	ca4f	bb4c	trmnok	Handles bad input data
-	ca59	bb56	getdtl	+
ca7f	ca5d	bb5a	stcurl	(Store current line #)
ca83	ca61	bb5e	snerr4	Jump to "syntax error"
ca86	ca64	bb61	trmnol	+

1.0	2.0	4.0	LABEL	DESCRIPTION
ca8f	ca6d	bb6a	doagin	(Print "redo from start" and...)
ca9f	ca7d	bb7a	get	GET instruction
cab6	ca94	bb91	gettty	+
cac6	caa7	bba4	inputn	INPUT# instruction
cad6	cab7	bbb4	iodone	Restore input to keyboard
cad8	cab9	bbb6	iorele	+
cae0	cac1	bbbe	input	INPUT instruction
caf1	cad2	bbcd	notqti	+
caf8	cada	bbd5	getagn	(To prompt & receive input)
cb0c	caed	bbe8	bufful	+
-	-	bbf1	pthrti	+
cb17	cafa	bbf5	qinlin	Prompts for & receives INPUT
cb21	cb04	bbff	ginlin	+
cb24	cb07	bc02	read	READ instruction
-	cb0e	bc09	inpcon	+
cb2a	cb10	bc0b	inpcol	(Entry from bb9c - with A=40 for GET)
cb30	cb16	bc11	inloop	(Entry from bcb1)
cb5b	cb42	bc3d	qdata	+
cb64	cb4b	bc46	getnth	(To prompt & input)
cb67	cb4e	bc49	datbk	+
cb6b	cb52	bc4d	datbkl	(Entry from bcd7)
cb7c	cb66	bc61	setqut	+
cb88	cb72	bc6d	resetc	+
cb89	cb73	bc6e	nowget	+
cb94	cb7e	bc79	nowgel	+
cba0	cb8a	bc85	numins	+
cba8	cb92	bc8d	strdn2	(Entry from bc82)
cbb4	cb9e	bc99	trmok	+
cbcf	cbb9	bcb4	datlop	+
cbe8	cbd2	bccd	nowlin	+
cbf5	cbdf	bcda	varend	+
cc00	cbea	bce5	vary0	Print "extra ignored" if @ kbd. & find a separator
cc11	cbfb	bcf6	inprts	End of READ
cc12	cbfc	bcf7	exignt	Message "extra ignored"
cc23	cc0d	bd07	tryagn	" "?redo from start"
cc36	cc20	bd19	next	NEXT instruction
cc3c	cc26	bd1f	getfor	(Entry from bd81)
cc3f	cc29	bd22	stxfor	+
cc4a	cc34	bd2d	errgo5	(To "next without for error")
cc4c	cc36	bd2f	havfor	+
cc8f	cc76	bd6f	newsgo	+
cc92	cc79	bd72	loopdn	Check data format - (check type mismatch)
cca4	cc8b	bd84	frmnum	Jump to "frmevl"
cca7	cc8e	bd87	chknum	Check that current type is numeric
cca9	cc90	bd89	chkstr	" " " " " string - cks. "valtyp"
ccaa	cc91	bd8a	chkval	+
ccb0	cc97	bd90	chkok	+
ccb1	cc98	bd91	docstr	+
ccb3	cc9a	bd93	chkerr	Print "type mismatch error"
ccb5	cc9c	bd95	errgo4	+
ccb8	cc9f	bd98	frmevl	Formula evaluator - evaluates all formulas
ccbe	cca5	bd9e	frmevl	+
ccc3	ccaa	bda3	lpoper	(Entry from be27)
ccd2	ccb9	bdb2	tstop	+
ccd5	ccbc	bdb5	loprel	+
ccf1	ccd8	bdd1	endrel	+
cd0a	ccf1	bdea	qprec	+
cd13	ccfa	bdf3	doprec	+

1.0	2.0	4.0	LABEL	DESCRIPTION
cd14	ccfb	bdf4	negprc	+
cd21	cd08	be01	finre1	+
cd2b	cd12	be0b	finre2	+
cd33	cd1a	be13	qprec1	+
cd3a	cd21	be1a	doprel	+
cd4a	cd31	be2a	snerr5	+
cd4d	cd34	be2d	pushf1	+
cd52	cd39	be32	pushf	+
cd5d	cd44	be41	forpsh	+
cd72	cd59	be56	qop	+
cd75	cd5c	be59	qopgo	+
cd77	cd5e	be5b	qchnum	+
cd7e	cd65	be62	unpstk	+
cd80	cd67	be64	pulstk	Restore "arg" from stack (pushed evaluation)
cd9a	cd81	be7e	qoprts	+
cd9c	cd83	be80	unprts	+
cd9d	cd84	be81	eval	Evaluates numeric formulas
cda1	cd88	be85	eval0	+
cda6	cd8d	be8a	eval1	+
cda9	cd90	be8d	eval2	+
cdbc	cda3	bea0	pival	Binary.value of pi
cdc1	cda8	bea5	qdot	+
cdd1	cdb8	beb5	strtxt	Immediate strings handler
cdda	cdc1	bebe	strtx2	+
cde0	cdc7	bec4	eval3	+
cde8	cdcf	becc	notop	Eval - not
cdf7	cdde	bedb	eval4	+
ce05	cdec	bee9	parchk	Evaluate function within parentheses (frmeul)
ce0b	cdf2	beef	chkcls	Check for right parenthesis -)
ce0e	cdf5	bef2	chkopn	Check for left parenthesis - (
ce11	cdf8	bef5	chkcom	Check for a comma - ,
ce13	cdfa	bef7	synchr	Compare "txtptr" against A, if <> then...
ce1c	ce03	bf00	snerr	Print "syntax error" & exit
ce21	ce08	bf05	domin	Set up function for future evaluation
ce23	ce0a	bf07	gonprc	+
-	-	bf0c	cksmb0	Checksum byte for the $b000 rom
-	-	bf0d	isujmp	Jmp "isvar"
-	-	bf10	pabbo	Patches
			patchg	+
-	-	bf1d	pcth0	+
-	-	bf1e	pcth1	+
-	-	bf21	patchh	+
-	-	bf2e	patchi	+
ce28	ce0f	bf8c	isvar	Set up a variable name search
ce2a	ce11	bf8e	zz6	+
ce2b	ce12	bf8f	isuret	+
-	-	bfc1	isuds	ds$ test and handler
ce53	ce42	bfd3	strrts	+
ce54	ce43	bfd4	gooo	+
-	ce54	bfe5	gooooo	+
-	-	bffc	chkds	Check for DS variable
ce76	ce69	c003	gettim	Assign time to TI
ce82	ce75	c00f	qstatv	+
-	-	c01c	qdsav	+
ce90	ce82	c040	gomovf	+
ce97	ce89	co47	isfun	Dispatch and eval if it's a function (Set up function references)
cec1	ceb3	c071	iknorm	+

1.0	2.0	4.0	LABEL	DESCRIPTION
cec6	ceb8	c076	fingo	Place function dispatch address in "jumper" & go
ced6	cec8	co86	orop	OR instruction
ced9	cecb	c089	andop	AND instruction
cf06	cef8	cob6	dorel	"<" instruction
cf1e	cf10	c0ce	strcmp	+
cf46	cf38	c0f6	stasgn	+
cf4b	cf3d	c0fb	nxtcmp	+
cf51	cf43	c101	qcomp	+
cf56	cf48	c106	getcmp	+
cf62	cf54	c112	docmp	+
cf6b	cf5d	c11b	goflot	+
cf6e	cf60	c11e	dim3	Re-entry for multi dim array
cf71	cf63	c121	dim	DIM instruction
cf7b	cf6d	c12b	ptrget	Searches for a BASIC variable
cf80	cf72	c130	ptrgtl	+
cf82	cf74	c132	ptrgt2	+
cf8c	cf7e	c13c	interr	Jump to "Syntax error"
cf8f	cf81	c13f	ptrgt3	+
cf9f	cf91	c14f	issec	+
cfa0	cf92	c150	eatem	+
cfaa	cf9c	c15a	nosec	+
cfb4	cfa6	c164	notstr	+
cfc4	cfb6	c174	turnon	+
cfcb	cfbd	c17b	strnam	+
cfe1	cfd3	c18f	stxfnd	(Find a BASIC variable)
cfe3	cfd5	c191	lopfnd	+
cfed	cfdf	c19b	lopfn	+
-	-	c1ab	nxtptr	Move search to next table entry
cffb	cfed	c1ac	notit	+
d005	cff7	c1b6	isletc	(Test A, set flags)
d00e	d000	c1bf	islrts	+
d00f	d001	c1c0	notfns	Did not find variable - create a new one
d015	d007	c1c6	ldzr	+
d01c	d00c	c1cb	notevl	(Test for TI)
d02c	d01c	c1db	gobadv	(Jump to "syntax error")
d02f	d01f	c1de	qstavr	Test for ST
-	-	c1e6	qdsvar	Test for DS
d037	d027	c1f2	varok	Good usable variable
d04d	d03d	c208	noteve	+
d44a	d448	c21c	aryva2	+
d44e	d44c	c220	aryva3	+
d459	d457	c228	aryvgo	Search the arrays
d488	d488	c259	aryget	Move thru the array tables
d492	d492	c263	gogo	+
-	-	c281	gogol	+
d4d0	d4d0	c290	dvarts	+
-	-	c29d	arydon	+
d079	d069	c2b9	finptr	Logs basic variable location [Bump address by 2]
d083	d073	c2c3	finnow	+
d088	d078	c2c8	fmaptr	Array pointer subroutine(Set up array pointer)
d094	d084	c2d4	jsrgm	+
d099	d089	c2d9	n32768	Floating point binary value: -32768 (top of screen)
d09d	d08d	c2dd	intidx	Evaluate formula - result is positive integer
d0a3	d093	c2e3	posint	Convert floating binary to positive integer
d0a7	d09a	c2ea	ayint	Convert floating binary to integer
d0b4	d0a7	c2f7	nonono	Branch to "illegal quantity error"
d0b6	d0a9	c2f9	qintgo	Jmp "qint"
d0b9	d0ac	c2fc(-C4A7)	isary	Locates and/or creates arrays

1.0	2.0	4.0	LABEL	DESCRIPTION
d0c3	d0b6	c306	indlop	+
d104	d0f7	c347	lopfda	+
d110	d103	c353	lopfdv	+
d11f	d112	c362	nmaryl	+
d12d	d120	c370	bserr	Print"bad subscript error"
d130	d123	c373	fcerr	Print"illegal quantity error"
d132	d125	c375	errgo3	(Jump to error message)
d149	d13c	c38c	nptfdp	+
d15d	d150	c39f	notflt	+
d166	d159	c3a8	stomlt	+
d16f	d162	c3b1	loppta	+
d17f	d172	c3c1	notdim	+
d1a2	d195	c3e4	grease	(Check available memory space)
d1b1	d1a4	c3f3	zerita	+
d1b6	d1a9	c3f8	deccur	+
d1d3	d1c6	c415	getdef	+
d1db	d1ce	c41d	inlpnm	+
d1f1	d1e4	c433	bserr7	Jump to "bad subscript error"
d1f4	d1e7	c436	omerr1	Jump to "out of memory error"
d1f7	d1ea	c439	inlpn2	+
d1f7	d1eb	c43a	inlpn1	+
d209	d1fc	c44b	addind	+
d218	d20d	c45c	notfll	+
d21e	d213	c462	stomll	+
d232	d227	c476	dimrts	+
d233	d228	c477	umult	Integer arithmetic routines for multi-dim arrays
d23c	d231	c480	umultd	+
d246	d23b	c48a	umultc	+
d25f	d254	c4a3	umlcnt	+
d263	d258	c4a7	umlrts	+
d264	d259	c4a8	fre	FRE instruction
d26b	d260	c4af	nofref	(Do garbage collection)
d278	d26d	c4bc	givayf	Converts integer to floating binary
d285	d27a	c4c9	pos	POS instruction
d287	d27c	c4cb	sngflt	+
d28b	d280	c4cf	errdir	Is instruction type indirect only?
d290	d288	c4d7	errguf	Print "illegal direct"
d295	d28d	c4dc	def	DEF instruction; also evaluates FN
d2c3	d2bb	c50a	getfnm	(check FN syntax)
d2d6	d2ce	c51d	fndoer	Evaluates FN
d2fc	d2f2	c541	defstf	+
d333	d329	c578	deffin	+
d349	d33f	c58e	strd	STR$ instruction
d353	d349	c598	timstr	Make a string out of info at $01ff
d359	d34f	c59e	strini	Make a string at address in "facmo pointer"
d361	d357	c5a6	strspa	+
d36b	d361	c5b0	strlit	Scans and sets up string elements
d371	d367	c5b6	strlt2	+
d37b	d371	c5c0	strget	+
d388	d37e	c5cd	strfin	+
d38c	d382	c5d1	strfil	+
d38d	d383	c5d2	strfi2	+
d399	d38f	c5de	strst2	+
-	d399	c5e8	strcp	+
d3aa	d3a4	c5f3	putnew	Check string temps - place data in temps
d3b2	d3ac	c5fb	errgo2	-1=Print "formula too complex error"
d3b5	d3af	c5fe	putnw1	+
d3d2	d3ce	c61d	getspa	Builds string vectors

1.0	2.0	4.0	LABEL	DESCRIPTION
d3d4	d3d0	c61f	tryag2	+
d3df	d3db	c62d	tryag3	+
-	-	c63a	tryag4	+
d3e9	d3e5	c644	strfre	+
-	-	c65a	getrts	+
d3f4	d3f0	c65b	garbag	+
d404	d400	c66a	garba2	Does 'garbage collection' - packs strings
-	-	c67e	gloop	+
-	-	c68a	col00	+
-	-	c693	col00b	+
-	-	c69e	col00a	+
-	-	c6a9	col01	+
-	-	c6b2	col02	+
-	-	c6ce	glop1	+
-	-	c6d8	col02b	+
-	-	c6f0	col02a	+
-	-	c700	grbend	Jmp "endgrb"
-	-	c703	col03	+
-	-	c716	endgrb	Moves "frespc" to "fretop"
-	-	c71f	skip2	+
-	-	c724	skip2a	+
-	-	c726	movpnt	+
-	-	c730	mov00	+
-	-	c735	movtop	+
-	-	c73f	mov01	+
-	-	c744	setinx	+
-	-	c746	set00	+
d515	d517	c74f	cat	Concatenate two strings: "fac" and "txtptr"
d535	d537	c76f	sizeok	+
d552	d554	c78c	movins	(Store string)
d560	d562	c79a	movstr	+
d564	d566	c79e	movdo	+
d568	d56a	c7a2	movlp	+
d571	d573	c7ab	mvdone	+
d57a	d57c	c7b4	mvstrt	+
d57b	d57d	c7b5	frestr	(Discard unwanted string)
d57e	d580	c7b8	frefac	+
d582	d584	c7bc	fretmp	Frees up temporary string pointers
-	-	c7de	res00	+
-	-	c7f6	fre01	+
d5ad	d5af	c7fc	frepla	+
-	-	c7fe	fre02	+
d5b3	d5b5	c811	fretms	(Clean descriptor stack)
d5c3	d5c5	c821	frerts	+
d5c4	d5c6	c822	chrd	CHR$ instruction
d5d8	d5da	c836	leftd	LEFT$ instruction
d5de	d5e0	c83c	rleft	+
d5e4	d5e6	c842	rleftl	+
d5e5	d5e7	c843	rleft2	+
d5e6	d5e8	c844	rleft3	+
d5fd	d5ff	c85b	pulmor	+
d604	d606	c862	rightd	RIGHT$ instruction
d60f	d611	c86d	midd	MID$ instruction
d620	d622	c87e	mid2	+
d637	d63b	c897	pream	Used by right - (pull string data)
d654	d656	c8b2	len	LEN instruction
d65a	d65c	c8b8	len1	(Switch string to numeric)
d663	d665	c8c1	asc	ASC instruction

1.0	2.0	4.0	LABEL	DESCRIPTION
d670	d672	c8ce	gofuc	Jump to "illegal quantity error"
d673	d675	c8d1	gtbytc	Does a "chrget" and "getbyt"
d676	d678	c8d4	getbyt	Evaluate the formula and...
d679	d67b	c8d7	conint	(Get a single byte value and return it in X)
d685	d687	c8e3	val	VAL instruction
d8a5	d6a7	c903	val2	+
d6bb	d6bd	c918	st2txt	+
d6c3	d6c5	c920	valrts	+
d6c4	d6c6	c921	getnum	Evaluate formula and return integer value (0-65535) (Get two parameters for POKE or WAIT)
d6ca	d6cc	c927	combyt	+
d6do	d6d2	c92d	getadr	Convert "fac" to integer - place in "poker"
d6e6	d6e8	c943	peek	PEEK instruction
d6f3	d6fb	c94e	getcon	+
-	d6fe	c951	dosgfl	+
d6f9	d707	c95a	poke	POKE instruction
d702	d710	c963	fuwait	WAIT instruction
d711	d71f	c972	stordo	+
d715	d723	c976	waiter	+
d71d	d72b	c97e	zerrts	+
d71e	d72c	c97f	faddh	Add 1/2 to fpb value in fac
d725	d733	c986	fsub	Move memory to "arg" and...
d728	d736	c989	fsubt	"-" instruction: fac=fac-arg
d737	d76e	c998	fadd5	+
d73c	d773	c99d	fadd	Move memory to "arg" and...
d73f	d776	c9a0	faddt	"+" instruction: fac=fac+arg
d74c	d783	c9ad	faddc	+
d768	d79f	c9c9	fadda	+
d76c	d7a3	c9cd	faddl	+
d778	d7af	c9d9	fadd4	+
d784	d7bb	c9e5	subit	+
d7a7	d7de	ca08	fadflt	+
d7ac	d7e3	ca0d	normal	Normalize "fac" - results of addition & subtraction
d7b0	d7e7	ca11	norm3	+
d7cc	d803	ca2d	zerofc	"fac"=0
d7ce	d805	ca2f	zerofl	+
d7d0	d807	ca31	zeroml	Make sign positive
d7d3	d80a	ca34	fadd2	+
d7f2	d829	ca53	norm2	+
d7fe	d835	ca5f	norml	+
d80b	d842	ca6c	squeez	+
d80d	d844	ca6e	rndshf	+
d81b	d852	ca7c	rndrts	+
d81c	d853	ca7d	negfac	Complement "fac" entirely
d822	d859	ca83	negfch	Complement just the number in "fac"
d844	d87b	caa5	incfac	Increment "fac"
d852	d889	cab3	incfrt	+
d853	d88a	cab4	overr	Print "overflow error"
d858	d88f	cab9	mulshf	Shifer routines - (multiply a byte)
d85a	d891	cabb	shftr2	+
d86e	d8a5	cacf	shiftr	+
d87b	d8b2	cadc	shftr3	+
d881	d8b8	cae2	shftr4	+
d885	d8bc	cae6	rolshf	+
d88f	d8c6	caf0	shftrt	+
d891	d8c8	caf2	fone	Floating-point-binary constant: 1
d896	d8cd	caf7	logcn2	" " " " 2.34518945e-38
d8ab	d8e2	cb0c	sqr05	Floating binary value: 1/sqr(2)

1.0	2.0	4.0	LABEL	DESCRIPTION
d8b0	d8c7	cb11	sqr2@	" " " sqr(2)
d8b5	d8cc	cb16	negh1f	" " " 1/2
d8ba	d8f1	cb1b	log2	" " " ln(2)
d8bf	d8f6	cb20	log	LOG instruction
d8c6	d8fd	cb27	logerr	Jump to "illegal quantity error"
d0c9	d900	cb2a	logl	+
-	-	cb5a	mul1n2	+
d8fd	d934	cb5e	fmult	Multiply: "fac"="fac"*"arg"
d900	d937	cb61	fmultt	"*" instruction: "arg" and "fac" loaded
d92b	d965	cb8f	mltply	+
d930	d96a	cb94	mltpll	+
d933	d96d	cb97	mltpl2	+
d94f	d989	cbb3	mltpl3	+
d95d	d997	cbc1	multrt	+
d95e	d998	cbc2	conupk	Unpack memory into "arg"
d989	d9c3	cbed	muldiv	Check and adjust exps of FPB mult and div (Test & adjust "fac" & "arg")
d98b	d9c5	cbef	mldexp	+
d996	d9d0	cbfa	tryoff	+
d9a6	d9e0	cc0a	mldvex	(Handle overflow and underflow)
d9ac	d9e6	cc10	zeremv	+
d9b1	d9eb	cc15	goover	Jump to "overflow error"
d9b4	d9ee	cc18	mul10	Multiply: "fac"="fac"*10
d9bf	d9f9	cc23	finml6	+
d9ca	da04	cc2e	mul10r	+
d9cb	da05	cc2f	tenc	Floating point binary value of 10
d9d0	da0a	cc34	div10	Divide: "fac"="fac"/10
d9d9	da13	cc3d	fdivf	(Perform divide-by)
d9e1	da1b	cc45	fdiv	Unpack memory and divide - (do divide-into)
d9e4	da1e	cc48	fdivt	"/" instruction: "fac"="arg"/"fac"
d9fb	da35	cc5f	divide	+
da11	da4b	cc75	savquo	+
da1e	da58	cc82	qshft	+
da21	da5b	cc85	shfarg	+
da2f	da69	cc93	divsub	+
da4c	da86	ccb0	ld100	+
da50	da8a	ccb4	divnrm	+
da5c	da96	ccc0	dv0err	Print "division by zero error"
da61	da9b	ccc5	movfr	Move "res" to "fac"
da74	daae	ccd8	movfm	Move memory to "fac"
da99	dad3	ccfd	mov2f	(Pack "fac" into memory)
da9c	dad6	cd00	movlf	+
daa2	dadc	cd06	movvf	+
daa6	dae0	cd0a	movmf	(Round off "fac" and) move "fac" to memory
dace	db08	cd32	movfa	Move "arg" to "fac"
dad0	db0a	cd34	movfal	+
dad4	db0e	cd38	movfal	+
dade	db18	cd42	movaf	Move "fac" to "arg" with round-off
dae1	db1b	cd45	movef	(" " " " without ")
dae3	db1d	cd47	movafl	+
daec	db26	cd50	movrts	+
daed	db27	cd51	round	Round-off "fac"
daf5	db2f	cd59	incrnd	+
dafd	db37	cd61	sign	Extract sign from "fac" - place in A
db01	db3b	cd65	fcsign	+
db03	db3d	cd67	fcomps	+
db0a	db44	cd6e	signrt	+
db0b	db45	cd6f	sgn	SGN instruction

1.0	2.0	4.0	LABEL	DESCRIPTION
db0e	db48	cd72	float	Float the signed integer in "fac"
db16	db50	cd7a	floats	Float the signed number in "fac"
db1b	db55	cd7f	floatc	+
db21	db5b	cd85	floatb	+
db2a	db64	cd8e	abs	ABS instruction
db2d	db67	cd91	fcomp	Compare "arg" and "fac": A=1 if "arg"<"fac" (Compare "fac" to memory)
db2f	db69	cd93	fcompn	+
db64	db9e	cdc8	fcompc	+
db6a	dba4	cdce	fcompd	+
db6d	dba7	cdd1	qint	Floating to fixed conversion: "fac"=int("fac")
db81	dbbb	cde5	qishft	+
db8c	dbc6	cdf0	qintrt	+
db8d	dbc7	cdf1	qintl	+
db9e	dbd8	ce02	int	INT instruction
dbbb	dbf5	ce1f	clrfac	Fill all positions of "fac" with contents of A
dbc4	dbfe	ce28	intrts	+
dbc5	dbff	ce29	fin	Convert input string to floating-pt. value in "fac"
dbc9	dc03	ce2d	finzlp	+
dbd8	dc12	ce3c	qplus	+
dbdc	dc16	ce40	finc	+
dbdf	dc19	ce43	findgq	+
dbe1	dc1b	ce45	finl	+
dc00	dc3a	ce64	finecl	+
dc02	dc3c	ce66	finec	+
dc05	dc3f	ce69	fnedgl	+
dc07	dc41	ce6b	finec2	+
dc13	dc4d	ce77	findp	+
dc19	dc53	ce7d	fine	+
dc1b	dc55	ce7f	finel	+
dc24	dc5e	ce88	findiv	+
dc2d	dc67	ce91	finmul	+
dc34	dc6e	ce98	finqng	+
dc39	dc73	ce9d	negxqs	+
dc3c	dc76	cea0	findig	+
dc43	dc7d	cea7	findgl	+
dc50	dc8a	ceb4	finlog	(Get new ASCII digit)
dc63	dc9d	cec7	finedg	+
dc72	dcac	ced6	mlex10	+
dc80	dcba	cee4	mlexmi	+
dc85	dcbf	cee9	n0999	Floating binary constant: 99999999.90625
dc8a	dcc4	ceee	n9999	" " " 99999999.5
dc8f	dcc9	cef3	nmil	" " " 1**+9
-	-	cef8	cksmc0	Checksum byte $c000 ROM
dc94	dcce	cf78	inprt	Print current line number - (Print "in", then #)
dc9f	dcd9	cf83	linprt	Print integer - Hi in A, Lo in Y
dcac	dce6	cf90	strou2	Jump to "strout"
dcaf	dce9	cf93	fout	Convert "fac" to string ending in 0, address in A & Y
dcb1	dceb	cf95	foutc	+
dcb9	dcf3	cf9d	foutl	+
dcd2	dd0c	cfb6	fout37	("fac"="fac"*1**+9)
dcdb	dd15	cfbf	fout7	+
dcdd	dd17	cfc1	fout4	+
dcf3	dd22	cfcc	fout3	(Multiply/Divide by 10)
-	dd2d	cfd7	fout38	+
dcfa	dd34	cfde	fout9	+
dd01	dd3b	cfe5	fout5	+
dd04	dd3e	cfe8	bigges	(Convert "fac" to positive integer)

1.0	2.0	4.0	LABEL	DESCRIPTION
dd19	dd53	cffd	foutpi	+
dd1a	dd54	cffe	fout6	+
dd25	dd5f	d009	fout39	+
dd36	dd70	d01a	fout16	+
dd38	dd72	d01c	fout8	+
dd3a	dd74	d01e	foutim	Clock entry into "fout"
dd3c	dd76	d020	fout2	+
dd60	dd9a	d044	fout41	+
dd62	dd9c	d046	fout40	+
dd69	dda3	d04d	foutyp	+
dd84	ddbe	d068	stxbuf	+
dd96	ddd0	d07a	fouldy	+
dd98	ddd2	d07c	foutl1	+
dda5	dddf	d089	foutl2	+
ddb5	ddef	d099	foutl4	+
ddc1	ddfb	d0a5	foutl5	+
ddd6	de10	d0ba	foutl9	+
ddd9	de13	d0bd	foutl7	+
ddde	de18	d0c2	fout20	+
dde3	de1d	d0c7	fhalf	Floating binary value: 1/2
dde5	de1f	d0c9	zero	+
dde8	de22	d0cc	foutbl	Tables of powers of -10**: (2.86866289E+36)
de0d	de46	d0f0	fdcend	End of powers table
de24	de5e	d108	timend	" " time conversion tables
			sqr	SQR instruction
de2e	de68	d112	fpwrt	[up-arrow] instruction: "arg"**"fac"
de37	de71	d11b	fpwrtl	+
de51	de8b	d135	fpwrl	+
de67	dea1	d14b	negop	">" instruction - Negate the number in "fac"
de71	deab	d155	negrts	+
de72	deac	d156	logeb2	Floating binary value: log(E) in base 2
de77	deb1	d15b	expcon	LOG and exponent - floating binary tables
dea0	deda	d184	exp	EXP instruction (of "fac")
deb0	deea	d194	stold	+
debb	def5	d19f	gomldv	+
debe	def8	d1a2	expl	+
dece	df08	d1b2	swaplp	+
def3	df2d	d1d7	polyx	Polynomial evaluator (function series evaluation)
df09	df43	d1ed	poly	+
df0d	df47	d1f1	poly1	+
df1c	df56	d200	poly3	(Multiply "fac" * "bufptr")
df20	df5a	d204	poly2	+
df2d	df67	d211	poly4	+
df3d	df77	d221	rmulc	(RND constants)
df41	df7b	d225	raddc	+
df45	df7f	d229	rnd	RND instruction
df63	df9d	d247	qsetnr	(Calculate new random number: "fac"=rnd("fac")
df78	dfb2	d25c	rndl	(Scramble "fac" mantissa)
df88	dfc2	d26c	strnex	+
df9b	dfd5	d27f	gmovmf	+
df9e	dfd8	d282	cos	COS instruction
dfa5	dfdf	d289	sin	SIN instruction - uses "fac"
dfd7	e011	d2bb	sinl	(Gosub ">" routine)
dfda	e014	d2be	sin2	+
dfe7	e021	d2cb	sin3	+
dfee	e028	d2d2	tan	TAN instruction
e016	e050	d2fa	cosc	+
e01a	e054	d2fe	pi2	Floating binary value pi/2

1.0	2.0	4.0	LABEL	DESCRIPTION
e01f	e059	d303	twopi	Floating binary value 2*pi
e024	e05e	d308	fr4	Floating binary value 1/4
e029	e063	d30d	sincon	SIN tables - Floating binary values:-4.88193226e-38
e048	e08c	d32c	atn	ATN instruction - uses "fac"
e050	e094	d334	atn1	+
e05e	e0a2	d342	atn2	+
e071	e0b5	d355	atn3	+
e077	e0bb	d35b	atn4	+
e078	e0bc	d35c	atncon	(Constants: 5.79991803e-36)
e0b5	e0f9	d399	initat	Basic system initialization code - copy of "chrget"
e0bb	e0ff	d39f	chdgot	+
e0cc	e110	d3b0	chdrts	+
e0d2	e116	d3b6	init	BASIC cold start - initialization and memory test
e0e7	e131	d3c9	movchg	(Copy "chrget" to zero page)
e116	e15d	d400	loopmm	+
e122	e165	d408	loopm1	+
e131	e174	d417	usedec	+
e135	e178	d41b	isedef	+
e174	e1b7	d44b	words	Message - 'bytes free'
e180	e1c4	d458	fremes	Message - '### commodore basic ###'
e1e1	e1de	d472	lastwr	Last byte of BASIC interpreter+1
040f	fd11	d472	calle	Call entry to Machine-language monitor
0427	fd17	d478	brke	Break " " " " "
0439	fd2d	d491	b3	PC-1 for Break
0447	fd48	d4ac	b5	Print entry data
0457	fd56	d4ba	strt	User command input
0477	fd65	d4c9	st1	Input command line
0482	fd70	d4d4	s0	Lookup command
0484	fd72	d4d6	s1	+
0498	fd82	d4e6	s2	Loop for all commands
0482	fd88	d4ec	putp	Move "tmp0" to "pch","pc1"
04bb	fd93	d4f7	dm	Display memory routine: "ar"=# bytes, "tmp0"=address
048f	fd97	d4fb	dm1	Write N bytes
04cf	fda7	d50b	byte	Read & store byte unless space, or "tmpc"=0
04e1	fdb9	d51d	by3	Increment "tmp0" address
04e7	fdbf	d523	setr	Set to access registers
0637	fdca	d52e	spac2	Print 2 spaces
063a	fdcd	d531	space	" 1 space
04f2	fdd0	d534	crlf	" return + line-feed
04f7	fdd5	d539	inctmp	Increment where "tmp0" points, by one
0501	fddf	d543	setwr	+
0502	fde0	d544	cmds	Table of Monitor commands
050a	fde8	d54c	adrh	" " " " ' addresses-1: Hi bytes
0512	fdf0	d554	adrl	" " " " " : Lo "
051a	fdf8	d55c	regk	Register header display
-	fe15	d579	altrit	(Begin new memory display line with prompt)
052d	fe23	d587	dsplyr	R - display registers command
053c	fe25	d589	d2	(Read & print "regk" loop)
055f	fe58	d5bc	dsplym	M - display memory command
0587	fe6e	d5d2	dspl	Test for stop key and ...
05ac	fe91	d5f5	beqs1	Jump to "strt"
05af	fe94	d5f8	errs1	" " "erropr"
05b2	fe97	d5fb	altr	Alter registers
05bd	fea2	d606	a12	SYS8
-	feb4	d618	a13	Set to alter registers
05c2	feb9	d61d	altm	Alter memory - read address & data
05cc	fec3	d627	a4	+
05ce	fec5	d629	a5	+

1.0	2.0	4.0	LABEL	DESCRIPTION
05d6	fecd	d631	a9	+
05d8	fecf	d633	go	G - go command
05eb	fee2	d646	g1	Original or new value to SP
05fe	ff07	d66b	exit	X - exit command
069c	ff0e	d672	errl	Jump to "erropr"
069f	ff11	d675	ld	L - Load command - default is from cassette #1
-	ff22	d688	l1	+
-	ff2f	d695	l2	File name must be next
06ce	ff31	d697	l3	Read file name
0714	ff43	d6a9	l4	File name too long
071f	ff47	d6ad	l5	Is this a load?
-	ff4b	d6b1	l6	Not a load
-	ff57	d6bd	l7	Load error
-	ff5c	d6c2	l8	Is this a default load?
-	ff65	d6cb	l9	Bad syntax
-	ff6c	d6d2	l10	Device 0
-	ff70	d6d6	l11	Device 3
-	ff7d	d6e3	l12	Bad syntax
-	ff8a	d6f0	l13	Missing end address
-	ff9a	d700	l20	Skip space loop
-	ffa3	d709	l14	Missing [return] at end
0604	e76a	d717	wroa	Monitor routines: Write address from "tmp0" stores
060a	e76c	d719	wroal	+
0613	e775	d722	wrob	Write a byte - unpack A into 2 characters in X & A
0622	e784	d731	wrtwo	Write 2 characters from X & A
062b	e78d	d73a	ascii	(Convert low nybble to ascii)
0634	e794	d741	asc1	+
063f	e797	d744	t2t2	Swap "tmp0" with "tmp2"
0641	e799	d746	t2t21	+
064f	e7a7	d754	rdoa	Read hex addr: return Hi in "tmp0", Lo in "tmp0+1"
0656	e7ae	d75b	rdoa2	+
065d	e7b5	d762	rdexit	Exit read
065e	e7b6	d763	rdob	Read hex byte and return in A
0665	e7be	d76b	rdob1	Space?
0672	e7cb	d778	rdob2	Convert first character to hex
067e	e7d8	d785	rdob3	" second " " "
0685	e7e0	d78d	hexit	Input one hex digit to A
068f	e7ea	d797	hex09	Exit with hex value in A
0690	e7eb	d798	rdoc	Read character
049b	7f7	d7a4	errpor	Operator error restart
?	?	d7ac	synerr	To "syntax error"
-	-	d7af	record	RECORD instruction
-	-	d7db	numadr	+
-	-	d7e1	rexnex	+
-	-	d7fe	doner	+
-	-	d801	qtyerl	+
-	-	d804	chk1	Disk parameter checks
-	-	d808	chkerl	+
-	-	d80b	chk2	+
-	-	d818	chk3	+
-	-	d81d	chk4	+
-	-	d824	chk5	+
-	-	d82e	chk6	+
-	-	d838	tabld	Dummy disk control messages
-	-	d873	catlog	CATALOG & DIRECTORY instructions
-	-	d87d	catalg	+
-	-	d889	catbld	+
-	-	d8a5	wg220	+

1.0	2.0	4.0	LABEL	DESCRIPTION
-	-	d8d2	skipb	+
-	-	d8d8	wg250	+
-	-	d8fe	wg255	+
-	-	d905	wg240	+
-	-	d911	wg235	+
-	-	d912	wg230	+
-	-	d91a	suba	Output
-	-	d923	subb	+
-	-	d92e	subbr	+
-	-	d92f	entry0	Find spare secondary address
-	-	d931	entry1	+
-	-	d935	entry2	+
-	-	d93f	rfound	+
-	-	d942	dopen	DOPEN instruction
-	-	d94f	dopen2	+
-	-	d95b	reclck	+
-	-	d95d	leav	+
-	-	d970	leav1	+
-	-	d977	append	APPEND instruction
-	-	d984	dappen	+
-	-	d991	errch1	Get disk status
-	-	d995	getds	+
-	-	d9ab	echks	+
-	-	d9b3	eread	+
-	-	d9bf	loop1	+
-	-	d9cb	errend	+
-	-	d9d2	format	HEADER instruction
-	-	d9de	ferr0	+
-	-	d9e1	dforma	+
-	-	d9ea	fbuild	+
-	-	d9f5	fcont	+
-	-	d9f8	ferrs	+
-	-	da04	ferrp	+
-	-	da07	dclose	DCLOSE instruction
-	-	da11	dclse	+
-	-	da1b	dclall	+
-	-	da1f	dcllp	+
-	-	da30	dclbye	+
-	-	da31	bobrec	Set up disk record
-	-	da3d	drecg	+
-	-	da43	drebld	+
-	-	da65	colect	COLLECT instruction
-	-	da6b	dcolle	+
-	-	da7a	dcoll0	+
-	-	da7e	backup	BACKUP instruction
-	-	da87	berr0	+
-	-	da8a	bback	+
-	-	da91	dbacku	+
-	-	da98	trans	+
-	-	da9b	transl	+
-	-	daa7	copy	COPY instruction
-	-	dab7	copy2	+
-	-	dabd	copcon	+
-	-	dac0	copy3	+
-	-	dac7	concat	CONCAT instruction
-	-	dad4	rsfn	Insert command string values
-	-	dae1	rdfn	+
-	-	daea	rdmov	+

1.0	2.0	4.0	LABEL	DESCRIPTION
-	-	daf8	rdrtØ	+
-	-	daf9	rdrtl	+
-	-	dafd	rid	+
-	-	db0d	dsave	DSAVE instruction
-	-	db1a	dsave2	+
-	-	db23	savldØ	+
-	-	db32	savldl	+
-	-	db3a	dload	DLOAD instruction
-	-	db44	dloerr	+
-	-	db47	dload2	+
-	-	db55	rename	+
-	-	db5f	drenl	+
-	-	db66	scrtch	SCRATCH instruction
-	-	db6c	dscrat	+
-	-	db78	numscr	+
-	-	db87	numlp	+
-	-	db93	numprt	+
-	-	db98	numbye	+
-	-	db99	ddirec	Check Direct command
-	-	db9d	dxcrØ	+
-	-	db9e	rusure	Ask "are you sure?" - wait for reply
-	-	dba3	rusurl	+
-	-	dbab	resur2	+
-	-	dbcb	ansno	+
-	-	dbd5	ansyes	+
-	-	dbd6	ansbye	+
-	-	dbd7	baddis	Print "bad disk"
-	-	dbel	oldclr	Clear DS$ and ST
-	-	dbfl	oldcll	+
-	-	dbfa	sendp	Assemble disk command string
-	-	dbfc	sendpl	+
-	-	dc02	sendp2	+
-	-	dc0d	rxfnl	+
-	-	dc14	rxfn2	+
-	-	dc1b	rxrec	+
-	-	dc24	rxid	+
-	-	dc2b	rxwrt	+
-	-	dc34	rxdl	+
-	-	dc3d	rxd2	+
-	-	dc44	rxdd	+
-	-	dc46	rplce	+
-	-	dc4c	tranr	+
-	-	dc57	rwrt	+
-	-	dc60	rwrtl	+
-	-	dc67	rwrts	+
-	-	dc68	dospar	Parse BASIC DOS command
-	-	dc89	parsel	+
-	-	dc9f	parnxt	+
-	-	dcb2	next7	+
-	-	dcba	snerl	+
-	-	dcbd	logadr	+
-	-	dcd8	reclem	+
-	-	dcea	recoo	+
-	-	dcf8	recon	+
-	-	dd03	donel	+
-	-	dd06	namell	+
-	-	dd09	onl	+
-	-	dd0f	unitl	+

1.0	2.0	4.0	LABEL	DESCRIPTION
-	-	dd15	drv1	+
-	-	dd34	qtyer2	+
-	-	dd37	ident	+
-	-	dd41	idcon	+
-	-	dd45	next3	+
-	-	dd4b	next4	+
-	-	dd60	name1	+
-	-	dd77	loop6	+
-	-	dd85	namcon	+
-	-	dd96	delim1	+
-	-	dd9e	nxxx	+
-	-	dda8	next6	+
-	-	ddaf	next6a	+
-	-	ddb3	sner8	+
	-	ddb6	parse2	+
-	-	ddcd	sner2	+
-	-	ddd0	drv2	+
-	-	ddec	on2	+
-	-	ddf2	unit2	+
-	-	ddf8	name2	+
-	-	de0f	delim2	+
-	-	de20	sner3	+
-	-	de23	done	+
-	-	de27	qtyerr	+
-	-	de2c	on	Get Device number
-	-	de49	newnam	Get file name
-	-	de6c	lenchk	
-	-	de74	errlen	
-	-	de79	nxxt5	
-	-	de81	next5	
-	-	de82	nxx5	
-	-	de87	getval	Get small variable parameter
-	-	de8a	gtvl2	+
-	-	de8f	cont	+
-	-	de9a	numerc	+
-	-	de9d	cksumd	Checksum $d000 ROM
-	-	dea4	patch2	Message "*** commodore basic 4.0 ***"+[2 returns]
e1e1	e1de	e000	cint	(Register/screen initialization)
e1ed	e1e7	e009	px1	Clear locations $60-$f8
e250	e229	e04b	clsr	Initialize line pointers for [clr]
?	e22d	e04f	lps1	$f4 - $f8 = $83
?	e23b	e05d	lps2	+
?	e23c	e05e	lps3	+
e236	e248	e06a	lps4	-2=Clear screen
?	e257	e079	nxtd	Start new screen line; adj. pointers for preset line
e5db	e25d	e07f	stupt	+
e5fa	e27a	e09c	stupz	Cursor column #
e604	e284	e0a6	stupr	+
e27d	e285	e0a7	lp2	(Input from keyboard - Get character from keyboard buffer & move rest of buffer down)
e282	e28a	e0ac	lp1	Keyboard buffer
e294	e29a	e0bc	loop4	Print single character
e297	e29d	e0bf	loop3	Wait for kbd. input; echo to screen, exit on [return]
e2b0	e2b1	e0d3	lp21	Get character from buffer
e2bd	e2bd	e0df	lp23	RUN or LOAD
e2c8	e2c8	e0ea	lp22	+
e2d1	e2d0	e0f2	clp5	+
e2da	e2d9	e0fb	clp6	+

1.0	2.0	4.0	LABEL	DESCRIPTION
e2fa	e2f4	e116	loop5	(Input from screen)
e303	e2fc	e11e	lop5	Cursor column #
e309	e302	e124	lop51	Strip bits 6 & 7
e313	e30c	e12e	lop54	+
e319	e312	e134	lop52	+
e31d	e31b	e138	lop53	If necessary, toggle ('quote' mode flag)
e327	e31f	e141	clp2	+
e335	e32b	e153	clp2a	Print character in acc
e338	e32e	e156	clp21	+
e33a	e330	e158	clp1	Last key = cr
e348	e33e	e166	clp7	+
e349	e33f	e167	qtswc	'quote'
e355	e34b	e173	qtsw1	+
e356	e34c	e174	nxt33	+
e358	e34e	e176	nxt3	Screen [rvs] flag
e35d	e352	e17a	nc3	+
e35f	e354	e17c	nvs	# of keyboard [inserts] outstanding
e365	e35a	e182	nvs1	Write acc to screen
e368	e35d	e185	jsts	Cursor column #
e383	e377	e19f	jsts1	Last line
e38a	e37e	e1a6	loop2	+
e392	e386	e1ae	lop2	+
e397	e38b	e1b3	jstsx	+
e3a3	e395	e1bd	jsxb	+
e3a4	e396	e1be	jts2	+
e3aa	e39c	e1c4	scr1	Scroll screen
e3b1	e3a3	e1cd	jsts2	Line wrap table
ec34	e3b4	e1de	bkln	Set up new screen line - Max columns per screen line
e3d0	e3c0	e1ea	bkln1	+
e3db	e3c9	e1f3	ntcn2	+
e3ea	e3d8	e202	prt	Output character in A
e406	e3f3	e21d	njt1	+
e412	e3ff	e229	ntcn	Number of keyboard inserts outstanding
e419	e405	e230	cnc3x	+
e428	e415	e23f	bk1	+
e433	e420	e24a	bk2	+
e439	e426	e250	ntcn1	(If=0, not in quote mode)
e43d	e42a	e254	cnc3	+
e440	e42d	e257	nc3w	[reverse]?
e447	e433	e25d	nc1	[home]?
e44e	e43a	e264	nc2	[cursor-right]?
e453	e43f	e275	jp14	+
e461	e44d	e277	ncz2	+
e464	e450	e27a	ncx2	[cursor-down]?
e476	e462	e28c	jp13	+
e479	e465	e28f	prt1	+
e481	e46d	e297	prt2	Increment cursor line number
e48f	e47a	e2a4	nxtx	+
e497	e482	e2ac	nxtx1	+
e4a5	e490	e2ba	up5	(If=0, not in quote mode)
e4b9	e4a4	e2ce	ins3	(End of line?)
e4c0	e4ab	e2d5	ins1	Screen line length
e4c2	e4ad	e2d7	ins2	+
e4d5	e4c0	e2ea	up9	# of keyboard [inserts] outstanding
e4d9	e4c4	e2ee	up6	+
e4de	e4c9	e2f3	up2	[cursor-down]?
e4ee	e4d9	e303	up1	+
e4fe	e4e8	e312	up3	+

1.0	2.0	4.0	LABEL	DESCRIPTION
e50e	e4f8	e322	nxt2	[reverse]?
e517	e500	e32a	nxt6	[cursor-right]?
e52d	e516	e340	jlp2	+
e530	e519	e343	nxln	Check for & perform scrolling
e536	e51e	e348	nxln2	+
e53e	e526	e350	nxlnl	+
548	e52f	e359	nxtl	Clear screen flags to 0
e55a	e540	e369	scrol	Scroll to screen
e56d	e552	e37b	scrll	+
e574	e559	e382	mlpl	+
e58d	e572	e39b	mlp2	+
e599	e57e	e3a7	scrl4	+
e59d	e582	e3ab	scrl5	+
e5a9	e58c	e3b5	scrl3	+
e5ca	e5aa	e3d2	mlp4	+
e5cd	e5ad	e3d5	mlp41	Wait for TI interrupt
e5d8	e5b7	e3df	mlp42	+
e5db	e5ba	e3e2	newln	Start new screen line
e617	e5cc	e3f3	newlx	+
e619	e5ce	e3f5	newll	+
e627	e5da	e401	newla	+
e63b	e5ed	e414	nell	+
e648	e5fa	e421	blkln	+
e65d	e60d	e434	blkl	+
e66b	e61b	e442	puls	Main Interrupt entry -IRQ & BRK
e67b	e62b	e452	puls1	Indirect jump through IRQ vector
e685	e62e	e455	key	60hz hardware interrupt: clock, cursor, keyboard
?	e649	e470	key5	+
e6b0	e64d	e474	key4	+
e6d4	e66e	e495	key3	+
e6db	e674	e49b	kl24	+
e6de	e677	e49e	kl2	+
e6ea	e682	e4a9	kl23	+
e6f4	e68b	e4b2	kl25	+
e6f7	e68e	e4b5	kl22	Keyboard scan
e701	e698	e4bf	kll	+
?	e6a7	e4ce	ckisl	+
?	e6b4	e4db	spck	Key image
e708	e6b6	e4dd	ckut	+
e709	e6b7	e4de	ckit	+
e714	e6c2	e4e9	ckitl	+
e72c	e6d6	e4fd	knl	+
e739	e6e2	e509	keyf	+
		e50b	prend0	+
e67e	e6e4	e600	prend	(Exit from Interrupt)
?	e6f3	e606	dspp	+
e75c	e6f8	e60b	char	Keyboard encoding table; 10 rows of 8 values
e7bc	e748	e65b	ldtb2	Screen table; 25 lines 80 COL $E755
e7d5	e761	e674	runtb	Message: 'dL"*'+[return]+"run"+[return]
				Characters entered when [shift-run] is hit
-	-	e67d	cksume	Checksum $e000 ROM
e810	e810	e810	pial	Keyboard PIA: I/O port A & data direction register
e811	e811	e811	piall	Control Register A
e812	e812	e812	piak	I/O port B & data direction register
				=$ff, unless hitting certain keys:
				[rvs]=$fe, "["=$fd, [space]=$fb & "<"=$f7
e813	e813	e813	pias	Control register B - Cassette 1 motor
				$35=on, $3d=off

1.0	2.0	4.0	LABEL	DESCRIPTION
e820	e820	e820	ieei	IEEE PIA: I/O port A & data direction register
e821	e821	e821	ieeis	Control register A - set output line CA2 $34=low, $3c=high
e822	e822	e822	ieeo	I/O port B & data direction registers Output data - set to $ff before read port A
e823	e823	e823	ieeos	Control register B - set output line CB2 $34=low, $3c=high
e840	e840	e840	pia	VIA: I/O port B: $cf=Cassette #2 motor on, $df= " " " off Bit 1 is IEEE NFRD " 3 " " ATN
e841	e841	e841	sync	I/O port A with handshaking
e842	e842	e842	p2db	Data direction register for I/O port B
e843	e843	e843	p2da	" " " " " " A
e844	e844	e844	til	Lo byte: Read timer 1, counter; write to timer 1 latch & ...
			chtim	+
e845	e845	e845	tih	Hi byte: initiate count
e846	e846	e846	till	Lo byte: read timer 1 latch
e847	e847	e847	tilh	Hi "
e848	e848	e848	t2l	Read timer 2 counter lo byte, & reset interrupt Write to timer 2 lo byte Microsecond clock Sets tone in CB2 sound: 0=low, $ff=high
e849	e849	e849	t2h	Read timer 2 counter hi by.; write to timer 2 hi by. Resets interrupt Millisecond clock
e84a	e84a	e84a	sr	Serial I/O shift register Sets timbre & octave of CB2 sound: $0f, $33 & $55 are popular settings
e84b	e84b	e84b	acr	Auxiliary control register: $10 enables CB2 sound $00 allows normal I/O after CB2 sound
e84c	e84c	e84c	pcr	Peripheral control register: $0c sets graphic mode $0e " text "
e84d	e84d	e84d	ifr	Interrupt flag register
e84e	e84e	e84e	ier	" enable "
e84f	e84f	e84f	syncl	I/O port A without handshaking
-	-	e880	cr0	6545 Video interface chip - CRT controller Bits 0-6: Set left margin of screen Bit 7: Hi=interlaced scan mode
-	-	e881	cr1	Sets top margin of screen (8 bit registers "cr2"-"crf" follow)
-	-	eff0-3		6551 ACIA - Superpet only
-	-	eff4-5		6850 ACIA - " "
-	-	eff8		System latch - " " Bit 0: lo=6502 CPU, hi=6809 CPU " 1: lo=read only, hi=read/write " 3: diagnostic sense, hi allows warm reset of 6502
-	-	effc		Bank select latch - Superpet only Bits 0-3 set the active bank Active bank addressed at $9000-9fff Bit 7 should be hi only when accessing $eff8
f000	f000	f000	msl	F000-F0D1 File messages: "too many files"
			msgl	
f00e	f00e	f00e	ms2	File message: "file open"
f017	f017	f017	ms3	" " : "file not open"

1.0	2.0	4.0	LABEL	DESCRIPTION
f024	f024	f024	ms4	" " : "file not found"
f032	f032	f032	ms5	" " : [return] + "searching "
f039	f039	f039	ms20	" " : "ing "
f03d	f03d	f03d	ms6	" " : "for "
f041	f041	f041	ms7	" " : [return] + "press play "
f04d	f04d	f04d	ms8	" " : "& record "
f056	f056	f056	ms9	" " : "on tape #"
f05f	f05f	f05f	ms10	" " : [return] + ...
f060	f060	f060	ms22	" " : "load"
f064	f064	f064	ms11	" " : [return] + "writing "
f06d	f06d	f06d	ms21	" " : [return] + ...
f06e	f06e	f06e	ms12	" " : "verify"
f074	f074	f074	ms13	" " : "device not present"
f086	f086	f086	ms15	" " : "not input file"
f094	f094	f094	ms16	" " : "not output file"
f0a3	f0a3	f0a3	ms17	" " : [return] + "found "
f0aa	f0aa	f0aa	ms18	" " : [return] + "ok" + [shift return]
f0ae	f0ae	f0ae	ms19	" " : [return] + "ready." + [shift return]
-	-	f0b6	msg30	" " : [return] + "are you sure ?"
-	-	f0c5	msg31	" " : [return] + "? bad disk " + [return]
f0b6	f0b6	f0d2	talk	SYS15 - Send 'Talk' (to IEEE) with attention
f0ba	f0ba	f0d5	listn	(- - " 'Listen' " " " " - uses A)
f0bc	f0bc	f0d7	list1	(Send IEEE command character)
f0e7	f0e4	f0ff	list4	Wait for DAV n=lo
f0f1	f0ee	f109	isour	(Send byte to IEEE)- Test channel
f107	f103	f11e	isr1	Wait for NFRD in to become low
f111	f10d	f128	isr0	+
f116	f112	f12d	isr2	IEEE status
f121	f11d	f138	isr3	Set IEEE out control DAV to hi
f12c	f128	f143	secnd	SYS27 - Send secondary address with listen - uses A
f132	f12d	f148	scatn	IEEE channel test: release ATN
-	-	f151	errs3	Option: timeout or wait for IEEE response
-	-	f15b	errs4	+
f13b	f136	f165	errp0	Status="listener timeout"
f13d	f138	f167	err01	+
f142	f13d	f16c	errp7	Print "device not present" Status="file not found" or "end of tape"
f146	f141	f170	errp1	Timeout on read, clear control lines
f14b	f146	f175	er001	+
e7de	f156	f185	msg	(Print system message)-(Send canned file message)
f15b	f164	f193	tksa	SYS26 - Secondary address with talk - uses A (Send byte, clear control lines)
f161	f169	f198	tkatn	SYS29 - Release ATN after talk - uses A
f167	f16f	f19e	ciout	SYS19 - Handshake character out - uses A (Send normal, deferred, IEEE character) (Send character to IEEE-jb)
f171	f177	f1a6	ci2	+
f176	f17c	f1ab	ci4	+
-	-	f1ae	untlk	SYS18 - untalk: drop IEEE device - no reg. used
f17a	f17f	~~f1b6~~ F1AE		+
f17e	f183	f1b9	unlsn	SYS17 - Send unlisten: no registers used
f187	f18c	f1c0	acptr	SYS20 - Handshake in byte (Input) from IEEE - uses A
f194	f199	f1cd	acp00	Set timer 1 to max
f19c	f19e	f1d2	acp01	Bit 6=t1 int
f1b5	f1ba	f1ee	acp03	Get IEEE input, invert & save on stack
f1c0	f1c5	f1f9	acp05	+
f1cc	f1d1	f205	getin	SYS10 - Get a buffered character - uses A
f1df	f1e1	f215	basin	SYS8 - Input a byte from channel - uses A

1.0	2.0	4.0	LABEL	DESCRIPTION
f1f1	f1f0	f224	bn10	If input device in A=screen, find line length
f200	f1fd	f231	bn20	If tape, set up to input characters, else IEEE
-	f20f	f243	jtg35	+
f205	f215	f249	jtget	Tape control
f218	f225	f259	jtg10	Get byte from tape buffer
f227	f228	f25c	bn30	Get ST
f22c	f22e	f262	bn32	+
f22d	f22f	f263	bn35	Jump to get IEEE input
f230	f232	f266	bsout	SYS9 - Output a byte to channel - uses A
f241	f23d	f271	bo10	(If not IEEE)
f247	f243	f277	bo20	(Send to cassette)
f248	f244	f278	bo21	Character to tape buffer storage
f273	f264	f298	jtp10	Get tape buffer character
f277	f268	f29c	rstor	Put in character in second tape buffer address
f2a4	f26e	f2a2	clall	SYS11 - abort, not close, all files - no regs. used
f27d	f272	f2a6	clrchn	SYS7 - Restore default I/O devices
			clrch	+
f28b	f27b	f2af	jx750	+
f299	f284	f2b8	jx770	Initialize I/O to default values
f2ab	f28d	f2c1	jltlk	Find & set up file data
f2ae	f28f	f2c3	jx600	+
f2b8	f299	f2cd	jzl00	Move file table entries to device, command
f2c7	f299	f2dc	jzl01	+
f2c8	f2a9	f2dd	close	CLOSE (logical file) instruction
f2cd	f2ae	f2e2	clos5	SYS4 - Close (logical) file # in A
			fclose	+
f2d2	f2b3	f2e7	clos10	Move files from table
f307	f2e1	f315	jx120	+
f30a	f2e4	f318	jx150	Recover list # & reduce open files by 1
f329	f300	f334	jx170	+
f32a	f301	f335	stop1	SYS22 - Test [stop] key - uses no registers— A=0 if stop wanted
f338	f30e	f342	stop2	+
f339	f30f	f343	stop	Stop if [stop] key depr.- "stop" also a lbl. at $b7c6
f33f	f315	f349	spmsg	Send message if in direct mode
-	f31d	f351	txtst	Test if in direct mode
-	f321	f355	txtrt	+
-	f322	f356	ld15	Program LOAD subroutines
-	f326	f35a	ld20	If device #=0 or 3, print "syntax error"
	f352	f38c	ld30	+
	f355	f38f	ld40	Strip bit 1 from ST
f39a	f378	f3b3	ld50	+
f39c	f37a	f3b5	ld60	+
	f380	f3bb	ld64	Test ST for end of file
	-	f3c1	ld90	+
	f387	f3c6	ld65	Tape end address=start address
f3a5	f395	f3d4	ld100	+
f3ae	f39e	f3dd	ld112	+
f3b7	f3a7	f3e6	ld120	Jump to "file not found"
f3ba	f3aa	f3e9	ld150	+
f3bf	f3af	f3ee	ld170	+
f346	f3c2	f401	load	LOAD instruction
f34b	f3c6	f405	ld10	0=LOAD; 1=VERIFY
-	f3c9	f408	loadnp	Transfer BASIC start/end to Tape start/end address
f350	f3ce	f40d	ld11	Wait for key switch change
f3db	f3e6	f425	ld209	Print "load error"
f3e5	f3ef	f42e	ld210	COLD START of BASIC - Reset to start & print "ready."
		f443	ld205	+

1.0	2.0	4.0	LABEL	DESCRIPTION
f3ff	f40a	f449	ld300	Print "searching"
f411	f41d	f45c	ld105	Print " file name"
f417	f423	f462	ld110	+
f421	f42d	f46c	ld115	+
f422	f42e	f46d	ld400	Print "loading" or "verifying"
f42b	f436	f475	ld410	"Verify"
f433	f43e	f47d	pars1	Initialize default values for I/O device
f43f	f447	f486	-	Get LOAD/SAVE type parameters
f45b	f45f	f49e	pro60	+
f45c	f460	f49f	pr070	Get expression from input buffer; put in X
f462	f466	f4a5	openi	Open IEEE channel for output
f475	f475	f4b4	openib	+
f47d	f47c	f4bb	op37	Abort IEEE & print "device not present"
f482	f481	f4c0	op35	+
f488	f487	f4c6	op40	+
f492	f491	f4d0	op45	+
f495	f494	f4d3	faf	Find specific tape header block
f4a3	f4a1	f4e0	faf20	+
f4b9	f4b5	f4f4	faf30	+
f4ba	f4b6	f4f5	faf40	+
f4bb	f4b7	f4f6	ver	VERIFY instruction
f4cf	f4c9	f508	ver10	Print "ok"
f4d4	f4ce	f50d	pars2	Get Open/Close parameters
f4f6	f4ef	f52e	pr100	+
f4fe	f4f7	f536	frl11	+
f504	f4fd	f53c	p200	Read file name - get string
f515	f50e	f54d	pr140	Check for end
f51c	f515	f554	pr147	+
f51d	f516	f555	pr150	Check for comma ","
f522	f519	f558	pr130	+
f527	f51e	f55d	pr135	To "syntax error"
f52a	f521	f560	open	OPEN instruction: from input parameters
f52d	f524	f563	op94	SYS3 - Open logical file - uses no registers
-	f526	f565	fopen	-2=Open file with preset parameters
f539	f528	f567	op98	+
f53d	f52f	f56e	op100	+
f566	f559	f598	op150	+
f579	f56e	f5ad	op160	Print "file not found" & then clear I/O
f57b	f570	f5af	ermsg	Send error message
f58b	f583	f5c2	op170	Search tape for header block
f592	f58a	f5c9	op200	Wait for cassette record/play switch
f59a	f592	f5d1	op171	+
f5aa	f5a3	f5e2	op172	+
f5ad	f5a5	f5e4	op175	+
f5ae	f5a6	f5e5	fah	Find any tape header block
f5b2	f5a9	f5e8	fah30	Read data record from tape
f5c5	f5bc	f5fb	fah50	+
f5d1	f5cb	f608	fah55	+
f5db	f5d3	f612	fah45	+
f5dd	f5d5	f614	fah40	Recover load/verify indicator
-	f5da	f619	tapeh	Write tape header
f5e7	f5ef	f62e	blnk2	Fill cassette buffer with spaces
f61e	f613	f652	th20	+
f632	f625	f664	th30	+
f64d	f63c	f67b	ldad2	Get start & end address from tape header for LOAD
f654	f643	f682	ldad3	Tape buffer to pointer
f667	f656	f695	-	Set tape buffer start address
f67c	f66b	f6aa	zz10	+

1.0	2.0	4.0	LABEL	DESCRIPTION
f67d	f66c	f6ab	ldadl	Set tape buffer start & end pointers
f695	f684	f6c3	sys	SYS instruction
-	f68d	f6cc	sv60	Set tape write start & end
f69e	f69c	f6dd	save	SAVE instruction
f6a1	f6a1	f6e0	sv3	Transfer BASIC start/end to tape
f6b1	f6a4	f6e3	sv5	Test for proper output device
f6b5	f6a8	f6e7	sv10	Abort IEEE, print "device not present"
f6ba	f6ad	f6ec	sv20	Screen the current device?
f6ce	f6d8	f717	sv30	Is current address = end address?
f6e3	f6ed	f72c	sv50	Send UNLISTEN to IEEE
f6e6	f6f0	f72f	clsei	Close IEEE channel
f6f6	f703	f742	sv100	+
f708	f716	f755	sv105	Write tape header
f736	f729	f768	udtim	Update jiffy clock
				(Hardware interrupt: cursor, tape & keyboard)
f743	f731	f770	ud10	+
f74e	f73b	f77a	ud20	+
f75b	f745	f784	ud30	+
f75d	f747	f786	ud40	+
f76c	f755	f794	ud45	+
f774	f75c	f79b	ud50	Zero the correction clock
f77c	f762	f7a1	ud60	Wait till keyboard input row changes
f787	f76c	f7ab	ud65	+
f76d	f788	f7ac	ud70	+
f78b	f770	f7af	chkin	SYS5 - open channel for input - uses X
				Set input device from logical file number
f79b	f77f	f7be	jx300	Abort IEEE & print "file not open"
f79d	f781	f7c0	jx305	+
f7a0	f784	f7c3	jx310	+
f7b5	f79b	f7da	jx320	Set input device to keyboard
f7dc	f7bc	f7df	jx330	Set output from logical file number
				Restore A,X,Y from stack & RTS
f7bf	f7a4	f7e3	jx3301	+
f7cd	f7ae	f7f0	jx340	+
f7d0	f7b1	f7f3	jx350	Check ST variable
f7dc	f7bc	f7fe	chkout	SYS6 - open channel for output - uses X
				Set output device from logical file number
			ckout	+
f7f8	f7d6	f818	ck10	To screen?
f806	f7e6	f828	jx360	Set CMD device
f80c	f7eb	f82d	jx370	Remember IEEE device #
f80d	f7ec	f831	jx3701	Test IEEE channel
f81e	f7f8	f83d	jx380	+
f821	f7fb	f840	jx390	Check ST variable
f82d	f806	f84b	jtp20	Increment tape buffer pointer
f83b	f812	f857	cstel	Print "press play" - Wait till cassette PLAY key hit
f842	f819	f85e	cs30	+
f851	f828	f86d	cs40	+
f85e	f835	f87a	cs10	Test cassette switch (Sense tape switch)
f868	f83e	f883	cs20	Bit 1 = cassette #1 read control
f870	f846	f88b	cs25	+
f871	f847	f88c	cste2	Ask for RECORD + PLAY, wait for PLAY
f87f	f855	f89a	rblk	Initiate tape read to buffer (Read tape to buffer)
f88a	f85e	f8a3	trd	(Read tape)
f8b4	f87f	f8c4	trd2	Toggle cassette #1 read control
-	f882	f8c7	trd3	+
f8b9	f886	f8cb	wblk	Initiate tape write (Write tape from buffer)
f8bc	f88c	f8ce	twrt	Checksum

1.0	2.0	4.0	LABEL	DESCRIPTION
f8c1	f88e	f8d3		+
f8c4	f890	f8d5	twrt2	-2=Write tape, leader length in A
f8cf	f89b	f8e0	tape	Common tape I/O
f8eb	f8b6	f8fb	tp20	+
-	f8c0	f905	tp30	Interblock delay loop
-	f8c2	f907	tp32	+
-	f8c4	f909	tp35	+
f8fa	f8ce	f913	tp40	Test, IRQ not modified
f912	f8e5	f92a	tp50	+
f913	f8e6	f92b	twait	Test if (Wait for) I/O complete or [stop] key down
f91e	f8f0	f935	tstop	Test stop key
f92b	f8fd	f942	stop3	BRK to immediate mode
f92e	f900	f945	stt1	Tape bit timing adjust
f943	f915	f95a	stt2	+
f94a	f91c	f961	stt3	+
f95f	f931	f976	read	Interrupt routine for tape read - read tape bits
f96d	f93f	f984	rd1	+
f97f	f951	f996	rads	+
f981	f953	f998	rd3	+
f995	f966	f9ab	jradj	+
f998	f969	f9ae	rjdj	+
f9b8	f988	f9cd	jrad2	+
f9bc	f98b	f9d0	srer	+
f9c5	f993	f9d8	radx2	+
f9cb	f997	f9dc	rad1	Right start bit/read bit sequence error
f9ce	f999	f9de	rad5	+
f9e5	f9ac	f9f1	rdbk	+
f9fa	f9bf	fa04	radp	+
fa04	f9c8	fa0d	radbk	Interrupt return
fa07	f9cb	fa10	rad3	+
fa12	f9d5	fa1a	rout2	+
fa14	f9d7	fa1c	rout1	+
fa29	f9ea	fa2f	rad3g	+
fa2e	f9ee	fa33	rad4	+
fa4a	fa07	fa4c	rad2	EOT bit received
fa53	fa0f	fa54	rad2y	+
fa5b	fa16	fa5b	rad2x	Timing constant
fa6c	fa26	fa6b	radq	+
fa70	fa2a	fa6f	radq1	+
fa84	fa3b	fa80	radq2	+
fa94	fa4a	fa8f	radk	+
fa9b	fa50	fa95	radr1	+
faa0	fa54	fa99	rdbk2	+
faa3	fa57	fa9c	radj	Read tape characters: initialize tape flags
fab5	fa67	faac	rd15	+
facc	fa7d	fac2	rd12	+
fad1	fa81	fac6	rd10	+
fad4	fa84	fac9	rd20	+
faeb	fa9a	fadf	rd22	+
faf2	faa0	fae5	rd200	+
fb06	fab1	faf6	rd40	+
fb0d	fab7	fafc	rd60	+
fb1b	fac5	fb0a	rd70	+
fb3a	fae2	fb27	rd80	+
fb45	faec	fb31	rd56	+
fb5a	faff	fb44	rd58	+
fba7	fb27	fb6c	rd52	+
fb8b	fb2b	fb70	rd55	Read character error

1.0	2.0	4.0	LABEL	DESCRIPTION
fb92	fb32	fb77	rd59	+
fb9d	fb3c	fb81	rd90	+
fba5	fb44	fb89	rd160	End
fba7	fb46	fb8b	rd161	+
fbb1	fb4f	fb94	rd167	Right leader count
fbbf	fb5b	fba0	rd175	+
fbd9	fb73	fbb8	rd180	Interrupt return
fbdc	fb76	fbbb	rd300	Reset tape read address (Reset tape I/O pointer)
fbe5	fb7f	fbc4	udst	Flag error into ST variable
fbdc	fb84	fbc9	newch	New character - reset counters for new byte
fc00	fb93	fbd8	write	Write a bit to tape
fc07	fb9a	fbdf	wrtw	+
fc09	fb9c	fbe1	wrt1	+
fc0b	fb9e	fbe3	wrtx	+
fc1c	fbaf	fbf4	wrt13	Tape write
fc21	fbb4	fbf9	wrtn	Tape write interrupt
fc48	fbd7	fc1c	wrtn2	+
fc65	fbf0	fc35	wrt3	Interrupt return
fc68	fbf3	fc38	wrt2	Cycle counter=0
fc74	fbfd	fc42	wrts	+
fc81	fc09	fc4e	wrts1	+
fc8c	fc13	fc58	wrt61	+
fc90	fc17	fc5c	wrt6	+
fc9f	fc26	fc6b	wrt7	+
fcb1	fc38	fc7d	wrt4	Tape character parity
fcb8	fc3e	fc83	wrtbk	+
fcbb	fc41	fc86	wrtn1	Write tape leader
		fc86	wrnc	+
fcc2	fc48	fc8d	wrend	+
fccf	fc54	fc99	wrtz	IRQ entry for print
fcfb	fc7b	fcc0	tnif	Terminate tape - restore interrupt vector
fd16	fc96	fcdb	stky	Right blocks completed
fd1b	fc9b	fce0	bsiv	IRQ fixer
?	fca6	fceb	tnof	Turn off both tape motors
fd7c	fcb4	fcf9	vprty	Checksum calculation
fd8a	fcbe	fd05	vpl0	+
fd90	fcc6	fd0b	wrt62	Advance load/save pointer
fcd0	fcc5	fd15	wrt64	+
fd38	fcd1	fd16	start	RESET: Power-on BASIC 1.0 diagnosticsas begin at $fd48
-	fcfe	fd49	nmi	NMI vectors through here
?	fd01	fd4c	bsit	Table of interrupt vectors
-	-	fd5c	cksumf	Checksum for $f000 ROM
-	-	ff93	concat	Jump Table from here on: CONCAT vector
-	-	ff96	dopen	DOPEN vector
-	-	ff99	dclose	DCLOSE "
-	-	ff9c	record	RECORD "
-	-	ff9f	format	HEADER "
-	-	ffa2	colect	COLLECT "
-	-	ffa5	backup	BACKUP "
-	-	ffa8	dcopy	COPY "
-	-	ffab	append	APPEND "
-	-	ffae	dsave	DSAVE "
-	-	ffb1	dload	DLOAD "
-	-	ffb4	dircat	DIRECTORY "
			dcat	CATALOG "
-	-	ffb7	rename	RENAME "
-	-	ffba	scratc	SCRATCH "

1.0	2.0	4.0	LABEL	DESCRIPTION
-	-	ffbd	readds	DS & DS$ " (Get disk status)
ffc0	ffc0	ffc0	copen	OPEN "
ffc3	ffc3	ffc3	cclos	CLOSE "
ffc6	ffc6	ffc6	coin	Set input device vector
ffc9	ffc9	ffc9	coout	Set output device "
ffcc	ffcc	ffcc	clschn	Restore normal (default) i/o devices vector
			ccchn	Same
ffcf	ffcf	ffcf	cinch	Input character from current input device vector
?	c481	ffcf	inchr	Same
ffd2	ffd2	ffd2	outch	Output a character vector
ffd5	ffd5	ffd5	cload	LOAD vector
ffd8	ffd8	ffd8	csave	SAVE "
ffdb	ffdb	ffdb	cverf	VERIFY "
ffde	ffde	ffde	csys	SYS "
ffe1	ffe1	ffe1	iscntc	Test stop key vector
ffe4	ffe4	ffe4	cgetl	GET a character from current input device vector
ffe7	ffe7	ffe7	ccall	Abort all i/o channels vector
ffea	ffea	ffea		Update clock vector
-	fffa	fffa		NMI vector
fffc	fffc	fffc		Reset vector
fffe	fffe	fffe		IRQ vector